

[illegible][illegible]

```

RRRRRRRR XX XX DDDDDDD I I I I I SSSSSSS KK KK
RRRRRRRR XX XX DDDDDDD I I I I I SSSSSSS KK KK
RR RR RR XX XX DD DD II II SS SS KK KK
RR RR RR XX XX DD DD II II SS SS KK KK
RR RR RR XX XX DD DD II II SS SS KK KK
RRRRRRRR XX XX DD DD II II SS SS KKKKKK
RRRRRRRR XX XX DD DD II II SS SS KKKKKK
RR RR XX XX DD DD II II SS SS KK KK
RR RR XX XX DD DD II II SS SS KK KK
RR RR XX XX DD DD II II SS SS KK KK
RR RR XX XX DDDDDDD I I I I I SSSSSSS KK KK
RR RR XX XX DDDDDDD I I I I I SSSSSSS KK KK
...
...
...
...

LL LL I I I I I SSSSSSS
LL LL I I I I I SSSSSSS
LL II SS
LL II SS
LL II SS
LL II SS
LL II SSSSSS
LL II SSSSSS
LL II SS
LL II SS
LL II SS
LL II SS
LLLLLLLLLL I I I I I SSSSSSS
LLLLLLLLLL I I I I I SSSSSSS

```


K 14
16-Sep-1984 00:14:14
5-Sep-1984 14:21:51

VAX-11 FORTRAN V3.4-56
DISK\$VMSMASTER:[ERF.SRC]RXDISK.FOR;1

Page 1

```
0001 SUBROUTINE RXDISK (LUN)
0002 C
0003 C Version: 'V04-000'
0004 C
0005 C*****
0006 C*
0007 C* COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0008 C* DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0009 C* ALL RIGHTS RESERVED.
0010 C*
0011 C* THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0012 C* ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0013 C* INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0014 C* COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0015 C* OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0016 C* TRANSFERRED.
0017 C*
0018 C* THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0019 C* AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0020 C* CORPORATION.
0021 C*
0022 C* DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0023 C* SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0024 C*
0025 C*
0026 C*****
0027 C
0028 C
0029 C
0030 C AUTHOR BRIAN PORTER CREATION DATE 9-APR-1979
0031 C
0032 C++
0033 C Functional description:
0034 C
0035 C This module displays entries for the RX211 controller. The format
0036 C of the error log buffer is as follows.
0037 C
0038 C +-----+
0039 C | control and status register |
0040 C +-----+
0041 C | device data buffer register |
0042 C +-----+
0043 C | data path number |
0044 C +-----+
0045 C | data path register |
0046 C +-----+
0047 C | final map register |
0048 C +-----+
0049 C | previous map register |
0050 C +-----+
0051 C | special error register |
0052 C +-----+
0053 C |
0054 C |-- extended error information --|
0055 C |
0056 C +-----+
0057 C
```

L 14
16-Sep-1984 00:14:14
5-Sep-1984 14:21:51

VAX-11 FORTRAN V3.4-56
DISK\$VMSMASTER:[ERF.SRC]RXDISK.FOR;1

Page 2

Modified by:

V03-003 SAR0233 Sharon A. Reynolds, 28-Mar-1984
Changed the call to UCBSL_OWNUIC to ORBSL_OWNER.
V03-002 SAR0095 Sharon A. Reynolds, 20-Jun-1983
Changed the carriage control in the 'format' statements
for use with ERF.
V03-001 SAR0051 Sharon A. Reynolds, 13-Jun-1983
Removed brief/cryptic support.
v02-004 BP0004 Brian Porter, 23-NOV-1981
Minor edit.
v02-003 BP0003 Brian Porter, 22-OCT-1981
Added extended status processing. Added 'device attention'
support.
v02-002 BP0002 Brian Porter, 23-JUL-1981
Added different uba handling.
v02-001 BP0001 Brian Porter, 1-JUL-1981
Added call to DHEAD and LOGGER.

INCLUDE 'SRC\$:MSGHDR.FOR /NOLIST'
INCLUDE 'SRC\$:DEVERR.FOR /NOLIST'

byte	lun
INTEGER*4	RX2_CS
INTEGER*4	RX2_ES
INTEGER*4	UBA_REGS(4)
integer*4	special_error_register
integer*4	extended_registers(2)
integer*2	extended_status(4)
EQUIVALENCE	(RX2_CS,EMB\$DV_REGS(0))
EQUIVALENCE	(RX2_ES,EMB\$DV_REGS(1))
EQUIVALENCE	(UBA_REGS,EMB\$DV_REGS(2))
equivalence	(special_error_register,emb\$dv_regsav(6))
equivalence	(extended_registers,emb\$dv_regsav(7))
equivalence	(extended_registers,extended_status)
INTEGER*4	FIELD
INTEGER*4	COMPRESSC
INTEGER*4	COMPRESS4
integer*4	definitive_error_code
integer*4	word_count_register
integer*4	current_track_address_drive0
integer*4	current_track_address_drive1
integer*4	target_track
integer*4	target_sector


```

0273 integer*4      soft_status
0274 integer*4      bad_track
0275
0276 logical*1      done
0277
0278 CHARACTER*20    RX_HEADSELCT(0:1)
0279 CHARACTER*7     RX_DENSITY(0:1)
0280
0281 CHARACTER*5     DRIVE_TYPE(0:1)
0282 DATA  DRIVE_TYPE(0)  /'RX01*'/
0283 DATA  DRIVE_TYPE(1)  /'RX02*'/
0284
0285 CHARACTER*17    V1RX2_CS(5:7)
0286 DATA  V1RX2_CS(5)    /'DONE*'/
0287 DATA  V1RX2_CS(6)    /'INTERRUPT ENABLE*'/
0288 DATA  V1RX2_CS(7)    /'TRANSFER REQUEST*'/
0289
0290 character*20    v2rx2_cs(8:9,0:1)
0291 data  v2rx2_cs(8,0)   /'SINGLE DENSITY*'/
0292 data  v2rx2_cs(8,1)   /'DOUBLE DENSITY*'/
0293 data  v2rx2_cs(9,0)   /'LOWER HEAD SELECTED*'/
0294 data  v2rx2_cs(9,1)   /'UPPER HEAD SELECTED*'/
0295
0296 CHARACTER*6     V3RX2_CS(15:15)
0297 data  v3rx2_cs(15)    /'ERROR*'/
0298
0299 CHARACTER*20    V1RX2_ES(0:4)
0300 DATA  V1RX2_ES(0)    /'CRC ERROR*'/
0301 DATA  V1RX2_ES(1)    /'SIDE 1 READY (RX03)*'/
0302 DATA  V1RX2_ES(2)    /'INITIALIZE DONE*'/
0303 DATA  V1RX2_ES(3)    /'DRIVE AC LO*'/
0304 DATA  V1RX2_ES(4)    /'DENSITY ERROR*'/
0305
0306 character*21    v2rx2_es(5:5,0:1)
0307 data  v2rx2_es(5,0)   /'SINGLE DENSITY DRIVE*'/
0308 data  v2rx2_es(5,1)   /'DOUBLE DENSITY DRIVE*'/
0309
0310 CHARACTER*13    V3RX2_ES(6:7)
0311 data  v3rx2_es(6)     /'DELETED DATA*'/
0312 data  v3rx2_es(7)     /'DRIVE READY*'/
0313
0314 CHARACTER*20    V4RX2_ES(10:11)
0315 data  v4rx2_es(10)    /'WORD COUNT OVERFLOW*'/
0316 data  v4rx2_es(11)    /'NON-EXISTENT MEMORY*'/
0317
0318 character*27    v1soft_status(0:0,0:1)
0319 data  v1soft_status(0,0) /'COMMAND WAS SINGLE DENSITY*'/
0320 data  v1soft_status(0,1) /'COMMAND WAS DOUBLE DENSITY*'/
0321
0322 character*26    v2soft_status(4:4,0:1)
0323 data  v2soft_status(4,0) /'DRIVE #0., SINGLE DENSITY*'/
0324 data  v2soft_status(4,1) /'DRIVE #0., DOUBLE DENSITY*'/
0325
0326 character*12    v3soft_status(5:5)
0327 data  v3soft_status(5)  /'HEAD LOADED*'/
0328
0329 character*26    v4soft_status(6:7,0:1)

```

```
0330      data      v4soft_status(6,0)      /'DRIVE #1., SINGLE DENSITY*'/
0331      data      v4soft_status(6,1)      /'DRIVE #1., DOUBLE DENSITY*'/
0332      data      v4soft_status(7,0)      /'DRIVE #0. SELECTED*'/
0333      data      v4soft_status(7,1)      /'DRIVE #1. SELECTED*'/
0334
0335      CALL FRCTOF (LUN)
0336
0337      call dhead1 (lun,'UBA RX211')
0338
0339      done = .false.
0340
0341      if (lib$extzv (5,1,rx2_cs) .eq. 1) done = .true.
0342
0343      CALL LINCHK (LUN,2)
0344
0345      20      WRITE(LUN,20) RX2_CS
0346      FORMAT(/' ',T8,'RX2CS',T24,Z8.4)
0347
0348      if (done) then
0349
0350      call linchk (lun,1)
0351
0352      25      write(lun,25) lib$extzv(4,1,rx2_cs)
0353      format(' ',t40,'DRIVE #',i1,'. SELECTED')
0354      endif
0355
0356      CALL OUTPUT (LUN,RX2_CS,V1RX2_CS,5,5,7,'0')
0357
0358      if (done) then
0359
0360      call output (lun,rx2_cs,v2rx2_cs,8,8,9,'2')
0361      endif
0362
0363      FIELD=LIB$EXTZV(11,1,RX2_CS)
0364
0365      CALL LINCHK (LUN,1)
0366
0367      45      WRITE(LUN,45) DRIVE_TYPE(FIELD)
0368      FORMAT(' ',T40,'CONTROLLER DRIVE TYPE ',
0369      1 A<COMPRESSC (DRIVE_TYPE(FIELD))>>)
0370
0371      CALL OUTPUT (LUN,RX2_CS,V3RX2_CS,15,15,15,'0')
0372
0373      CALL LINCHK (LUN,1)
0374
0375      55      WRITE(LUN,55) RX2_ES
0376      FORMAT(' ',T8,'RX2ES',T24,Z8.4)
0377
0378      if (done) then
0379
0380      if (lib$extzv(1,3,rx2_cs) .eq. 7) then
0381
0382      call linchk (lun,1)
0383
0384      definitive_error_code = lib$extzv(0,7,rx2_es)
0385
0386      if (definitive_error_code .eq. '010'o) then
```



```
0387
0388
0389 60  write(lun,60)
0390      format(' ',t40,'DRIVE #0., 'HOME' FAILURE ON INIT')
0391
0392      else if (definitive_error_code .eq. '020'o) then
0393
0394 65  write(lun,65)
0395      format(' ',t40,'DRIVE #1., 'HOME' FAILURE ON INIT')
0396
0397      else if (definitive_error_code .eq. '040'o) then
0398
0399 70  write(lun,70)
0400      format(' ',t40,'TRIED TO ACCESS TRACK >77.')
0401
0402      else if (definitive_error_code .eq. '050'o) then
0403
0404 75  write(lun,75)
0405      format(' ',t40,'HOME' BEFORE DESIRED TRACK')
0406
0407      else if (definitive_error_code .eq. '070'o) then
0408
0409 80  write(lun,80)
0410      format(' ',t40,'2. REVOLUTIONS, SECTOR NOT FOUND')
0411
0412      else if (definitive_error_code .eq. '110'o) then
0413
0414 85  write(lun,85)
0415      format(' ',t40,'>40 MICRO-SEC, NO 'SEP' CLOCK')
0416
0417      else if (definitive_error_code .eq. '120'o) then
0418
0419 90  write(lun,90)
0420      format(' ',t40,'A PREAMBLE NOT FOUND')
0421
0422      else if (definitive_error_code .eq. '130'o) then
0423
0424 95  write(lun,95)
0425      format(' ',t40,'PREAMBLE BUT NO 'ID'')
0426
0427      else if (definitive_error_code .eq. '140'o) then
0428
0429 100 write(lun,100)
0430      format(' ',t40,'CRC ERROR, HEADER')
0431
0432      else if (definitive_error_code .eq. '150'o) then
0433
0434 105 write(lun,105)
0435      format(' ',t40,'HEADER/DESIRED TRACK MISMATCH')
0436
0437      else if (definitive_error_code .eq. '160'o) then
0438
0439 110 write(lun,110)
0440      format(' ',t40,'TOO MANY TRIES FOR 'IDAM'')
0441
0442      else if (definitive_error_code .eq. '200'o) then
0443      write(lun,115)
```



```
0444 115  format(' ',t40,'CRC ERROR, DATA')
0445
0446      else if (definitive_error_code .eq. '220'o) then
0447
0448      write(lun,120)
0449 120  format(' ',t40,'DIAGNOSTIC MODE TEST FAILURE')
0450
0451      else if (definitive_error_code .eq. '240'o) then
0452
0453      write(lun,125)
0454 125  format(' ',t40,'DENSITY ERROR')
0455
0456      else if (definitive_error_code .eq. '250'o) then
0457
0458      write(lun,130)
0459 130  format(' ',t40,'SET DENSITY, INCORRECT 'KEYWORD'')
0460      endif
0461      else
0462
0463      CALL OUTPUT (LUN,RX2_ES,V1RX2_ES,0,0,4,'0')
0464
0465      call output (lun,rx2_es,v2rx2_es,5,5,5,'2')
0466
0467      CALL OUTPUT (LUN,RX2_ES,V3RX2_ES,6,6,6,'0')
0468      endif
0469
0470      call output (lun,rx2_es,v3rx2_es,6,7,7,'0')
0471
0472      CALL LINCHK (LUN,1)
0473
0474      write(lun,25) lib$extzv (8,1,rx2_es)
0475
0476      CALL OUTPUT (LUN,RX2_ES,V3RX2_ES,10,10,11,'0')
0477      endif
0478
0479      call linchk (lun,4)
0480
0481      write(lun,135) (extended_status(i),i = 1,4)
0482 135  format(' ',t8,'EXTENDED STATUS',4(t28,z4.4,:/))
0483
0484      definitive_error_code = lib$extzv(0,8,extended_registers(1))
0485
0486      call linchk (lun,1)
0487
0488      if (definitive_error_code .eq. '010'o) then
0489
0490      write(lun,60)
0491
0492      else if (definitive_error_code .eq. '020'o) then
0493
0494      write(lun,65)
0495
0496      else if (definitive_error_code .eq. '040'o) then
0497
0498      write(lun,70)
0499
0500      else if (definitive_error_code .eq. '050'o) then
```



```
0501      write(lun,75)
0502
0503      else if (definitive_error_code .eq. '070'o) then
0504      write(lun,80)
0505
0506      else if (definitive_error_code .eq. '110'o) then
0507      write(lun,85)
0508
0509      else if (definitive_error_code .eq. '120'o) then
0510      write(lun,90)
0511
0512      else if (definitive_error_code .eq. '130'o) then
0513      write(lun,95)
0514
0515      else if (definitive_error_code .eq. '140'o) then
0516      write(lun,100)
0517
0518      else if (definitive_error_code .eq. '150'o) then
0519      write(lun,105)
0520
0521      else if (definitive_error_code .eq. '160'o) then
0522      write(lun,110)
0523
0524      else if (definitive_error_code .eq. '200'o) then
0525      write(lun,115)
0526
0527      else if (definitive_error_code .eq. '220'o) then
0528      write(lun,120)
0529
0530      else if (definitive_error_code .eq. '240'o) then
0531      write(lun,125)
0532
0533      else if (definitive_error_code .eq. '250'o) then
0534      write(lun,130)
0535      endif
0536
0537      word_count_register = lib$extzv(8,8,extended_registers(1))
0538
0539      call linchk (lun,1)
0540
0541      write(lun,137) 'WORD COUNT REGISTER ',word_count_register,' (HEX)'
0542      format(' ',t40,a,z4.4,a)
0543
0544      current_track_address_drive0 = lib$extzv(16,8,extended_registers(1))
0545
0546
0547
0548
0549
0550
0551
0552
0553
0554
0555
0556
0557
```

137


```
0558      call linchk (lun,1)
0559
0560      write(lun,140) current_track_address_drive0
0561 140    format(' ',t40,'CURRENT TRACK #',
0562            1 i<compress4 (current_track_address_drive0)>,'.., DRIVE #0.')
0563
0564      current_track_address_drive1 = lib$extzv(24,8,extended_registers(1))
0565
0566      call linchk (lun,1)
0567
0568      write(lun,145) current_track_address_drive1
0569 145    format(' ',t40,'CURRENT TRACK #',
0570            1 i<compress4 (current_track_address_drive1)>,'.., DRIVE #1.')
0571
0572      target_track = lib$extzv(0,8,extended_registers(2))
0573
0574      call linchk (lun,1)
0575
0576      write (lun,150) target_track
0577 150    format(' ',t40,'TRACK #',i<compress4 (target_track)>,
0578            1 '.., TARGET TRACK')
0579
0580      target_sector = lib$extzv(8,8,extended_registers(2))
0581
0582      call linchk (lun,1)
0583
0584      write(lun,155) target_sector
0585 155    format(' ',t40,'SECTOR #',i<compress4 (target_sector)>,
0586            1 '.., TARGET SECTOR')
0587
0588      soft_status = lib$extzv(16,8,extended_registers(2))
0589
0590      call output (lun,soft_status,v1soft_status,0,0,0,'2')
0591
0592      call output (lun,soft_status,v2soft_status,4,4,4,'2')
0593
0594      call output (lun,soft_status,v3soft_status,5,5,5,'0')
0595
0596      call output (lun,soft_status,v4soft_status,6,6,7,'2')
0597
0598      if (definitive_error_code .eq. '150'o) then
0599
0600      bad_track = lib$extzv(24,8,extended_registers(2))
0601
0602      call linchk (lun,1)
0603
0604      write(lun,160) bad_track
0605 160    format(' ',t40,'SELECTED DRIVE AT TRACK #',
0606            1 i<compress4 (bad_track)>,'..')
0607      endif
0608
0609      if (emb$w_hd_entry .ne. 98) then
0610
0611      call uba_datapath (lun,uba_regs(1),uba_regs(2))
0612
0613      call uba_mapping (lun,-1,uba_regs(3))
0614
```


VAX-11 FORTRAN V3.4-56 Page 9
DISK\$VMSMASTER:[ERF.SRC]RXDISK.FOR;1

PROGRAM SECTIONS

Name	Bytes	Attributes
0 \$CODE	2609	PIC CON REL LCL SHR EXE RD NOWRT LONG
1 \$PDATA	962	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2 \$LOCAL	2040	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
3 EMB	512	PIC OVR REL GBL SHR NOEXE RD WRT LONG
Total Space Allocated	6123	

ENTRY POINTS

Address	Type	Name
0-00000000		RXDISK

VARIABLES

Address	Type	Name	Address	Type	Name
2-00000298	I*4	BAD_TRACK	2-00000284	I*4	CURRENT_TRACK_ADDRESS_DRIVED
2-00000288	I*4	CURRENT_TRACK_ADDRESS_DRIVE1	2-0000027C	I*4	DEFINITIVE_ERROR_CODE
2-00000277	L*1	DONE	3-0000001C	L*1	EMBSB_DV_CLASS
3-00000010	L*1	EMBSB_DV_ERTCNT	3-00000011	L*1	EMBSB_DV_ERTMAX
3-0000003E	L*1	EMBSB_DV_NAMLNG	3-0000003A	L*1	EMBSB_DV_SLAVE
3-0000001D	L*1	EMBSB_DV_TYPE	3-00000036	I*4	EMBSL_DV_CHAR
3-00000012	I*4	EMBSL_DV_IOSB1	3-00000016	I*4	EMBSL_DV_IOSB2
3-00000026	I*4	EMBSL_DV_MEDIA	3-0000004E	I*4	EMBSL_DV_NUMREG
3-0000002E	I*4	EMBSL_DV_OPCNT	3-00000032	I*4	EMBSL_DV_OWNUIC
3-0000001E	I*4	EMBSL_DV_RQPID	3-00000000	I*4	EMBSL_HD_SID
3-0000003F	CHAR	EMBST_DV_NAME	3-00000024	I*2	EMBSW_DV_BCNT
3-00000022	I*2	EMBSW_DV_BOFF	3-0000002C	I*2	EMBSW_DV_ERRCNT
3-0000003C	I*2	EMBSW_DV_FUNC	3-0000001A	I*2	EMBSW_DV_STS
3-0000002A	I*2	EMBSW_DV_UNIT	3-00000004	I*2	EMBSW_HD_ENTRY
3-0000000E	I*2	EMBSW_HD_ERRSEQ	2-00000278	I*4	FIELD
2-0000029C	I*4	I	AP-00000004	L*1	LUN
3-00000052	I*4	RX2_CS	3-00000056	I*4	RX2_ES
2-00000294	I*4	SOFT_STATUS	3-0000006A	I*4	SPECIAL_ERROR_REGISTER
2-00000290	I*4	TARGET_SECTOR	2-0000028C	I*4	TARGET_TRACK
2-00000280	I*4	WORD_COUNT_REGISTER			

ARRAYS

Address	Type	Name	Bytes	Dimensions
2-00000036	CHAR	DRIVE_TYPE	10	(0:1)
3-00000000	L*1	EMB	512	(0:511)
3-00000052	I*4	EMBSL_DV_REGSAV	420	(0:104)
3-00000006	I*4	EMBSQ_HD_TIME	8	(2)
3-0000006E	I*4	EXTENDED_REGISTERS	8	(2)
3-0000006E	I*2	EXTENDED_STATUS	8	(4)
2-00000028	CHAR	RX_DENSITY	14	(0:1)
2-00000000	CHAR	RX_HEADSELCT	40	(0:1)

3-0000005A	I*4	UBA_REGS	16	(4)
2-00000040	CHAR	V1RX2_CS	51	(5:7)
2-000000C9	CHAR	V1RX2_ES	100	(0:4)
2-00000199	CHAR	V1SOFT_STATUS	54	(0:0, 0:1)
2-00000073	CHAR	V2RX2_CS	80	(8:9, 0:1)
2-0000012D	CHAR	V2RX2_ES	42	(5:5, 0:1)
2-000001CF	CHAR	V2SOFT_STATUS	52	(4:4, 0:1)
2-000000C3	CHAR	V3RX2_CS	6	(15:15)
2-00000157	CHAR	V3RX2_ES	26	(6:7)
2-00000203	CHAR	V3SOFT_STATUS	12	(5:5)
2-00000171	CHAR	V4RX2_ES	40	(10:11)
2-0000020F	CHAR	V4SOFT_STATUS	104	(6:7, 0:1)

LABELS

Address	Label	Address	Label	Address	Label	Address	Label	Address	Label	Address	Label
1-00000068	20'	1-0000007B	25'	1-00000098	45'	1-000000BC	55'	1-000000CE	60'	1-000000F7	65'
1-00000120	70'	1-00000142	75'	1-00000165	80'	1-0000018D	85'	1-000001B2	90'	1-000001CE	95'
1-000001EA	100'	1-00000203	105'	1-00000228	110'	1-00000249	115'	1-00000260	120'	1-00000284	125'
1-00000299	130'	1-000002C1	135'	1-000002E4	137'	1-000002EF	140'	1-0000031A	145'	1-00000345	150'
1-0000036B	155'	1-00000393	160'	1-000003BD	165'						

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	Type	Name	Type	Name	Type	Name	Type	Name
I*4	COMPRESS4	I*4	COMPRESSC		DHEAD1		FRCTOF		IRP\$L_PID
	IRP\$W_BCNT		IRP\$W_BOFF	I*4	LIB\$EXTZV		LINCHK		IRP\$Q_IOSB
	RXDISK_QIO		UBA_DATAPATH		UBA_MAPPING		UCB\$B_ERTCNT		OUTPUT
	UCB\$L_MEDIA		UCB\$L_OPCNT		UCB\$W_ERRCNT		UCB\$B_ERTMAX		UCB\$L_CHAR

Subroutine RXDISK_QIO (lun,emb\$w_dv_func)

include 'src\$:qiocommon.for /nolist'

byte lun

integer*2 emb\$w_dv_func

integer*4 qiocode(0:1,0:63)

if (qiocode(0,0) .eq. 0) then

qiocode(1,08) = %loc(io\$_packack)

qiocode(1,11) = %loc(io\$_writepblk)

qiocode(1,12) = %loc(io\$_readpblk)

qiocode(1,26) = %loc(io\$_setchar)

qiocode(1,27) = %loc(io\$_sensechar)

qiocode(1,30) = %loc(io\$_format)

qiocode(1,32) = %loc(io\$_writelblk)

qiocode(1,33) = %loc(io\$_readlblk)

qiocode(1,35) = %loc(io\$_setmode)

qiocode(1,39) = %loc(io\$_sensemode)

qiocode(1,48) = %loc(io\$_writevblk)

qiocode(1,49) = %loc(io\$_readvblk)

qiocode(1,50) = %loc(io\$_access)

qiocode(1,51) = %loc(io\$_create)

qiocode(1,52) = %loc(io\$_deaccess)

qiocode(1,53) = %loc(io\$_delete)

qiocode(1,54) = %loc(io\$_modify)

qiocode(1,56) = %loc(io\$_acpcontrol)

qiocode(1,57) = %loc(io\$_mount)

do 10,i = 0,63


```
0321      qiocode(0,i) = 33
0322
0323      if (qiocode(1,i) .eq. 0) then
0324
0325      qiocode(1,i) = %loc(qio_string)
0326      endif
0327
0328      10      continue
0329      endif
0330
0331      call irp$w_func (lun,emb$w_dv_func,
0332      1 qiocode(0,lib$extzv(0,6,emb$w_dv_func)))
0333
0334      return
0335
0336      end
```

PROGRAM SECTIONS

Name	Bytes	Attributes
0 \$CODE	225	PIC CON REL LCL SHR EXE RD NOWRT LONG
1 \$PDATA	8	PIC CON REL LCL SHR NOEXE RD NOWRT LONG
2 \$LOCAL	548	PIC CON REL LCL NOSHR NOEXE RD WRT LONG
3 QIOCOMMON	1247	PIC OVR REL GBL SHR NOEXE RD WRT LONG
Total Space Allocated	2028	

ENTRY POINTS

Address	Type	Name
0-00000000		RXDISK_QIO

VARIABLES

Address	Type	Name	Address	Type	Name
AP-00000008a	I*2	EMB\$W_DV_FUNC	2-00000200	I*4	I
3-00000442	CHAR	IOS_ABORT	3-00000340	CHAR	IOS_ACCESS
3-000003C2	CHAR	IOS_ACPCONTROL	3-000004B3	CHAR	IOS_AVAILABLE
3-00000297	CHAR	IOS_CLEAN	3-00000369	CHAR	IOS_CREATE
3-00000385	CHAR	IOS_DEACCESS	3-00000393	CHAR	IOS_DELETE
3-0000026D	CHAR	IOS_DIAGNOSE	3-00000065	CHAR	IOS_DRVCLR
3-000004CB	CHAR	IOS_DSE	3-000000A9	CHAR	IOS_ERASETAPE
3-00000276	CHAR	IOS_FORMAT	3-00000071	CHAR	IOS_INITIALIZE
3-00000014	CHAR	IOS_LOADMCODE	3-000003A1	CHAR	IOS_MODIFY
3-000003E2	CHAR	IOS_MOUNT	3-00000000	CHAR	IOS_NOP
3-0000009D	CHAR	IOS_OFFSET	3-000000EB	CHAR	IOS_PACKACK
3-000000E0	CHAR	IOS_QSTOP	3-000003EF	CHAR	IOS_RDSTATS
3-00000421	CHAR	IOS_READCSR	3-00000169	CHAR	IOS_READHEAD
3-000002B6	CHAR	IOS_READLBLK	3-0000013F	CHAR	IOS_READPBLK

3-00000200	CHAR	IOS_READPRESET	3-00000195	CHAR	IOS_READTRACKD
3-0000033A	CHAR	IOS_READVBLK	3-0000045A	CHAR	IOS_READWTHBUF
3-00000484	CHAR	IOS_READWTHXBUF	3-0000004D	CHAR	IOS_RECAL
3-0000007C	CHAR	IOS_RELEASE	3-000001AB	CHAR	IOS_REREADN
3-000001B8	CHAR	IOS_REREADP	3-000000CA	CHAR	IOS_RETCENTER
3-000002E6	CHAR	IOS_REWIND	3-000002C9	CHAR	IOS_REWINDOFF
3-000000FC	CHAR	IOS_SEARCH	3-00000024	CHAR	IOS_SEEK
3-00000231	CHAR	IOS_SENSECHAR	3-00000309	CHAR	IOS_SENSEMODE
3-0000021D	CHAR	IOS_SETCHAR	3-000003B8	CHAR	IOS_SETCLOCK
3-00000088	CHAR	IOS_SETCLOCKP	3-000002DD	CHAR	IOS_SETMODE
3-000002ED	CHAR	IOS_SKIPFILE	3-000002FA	CHAR	IOS_SKIPRECORD
3-00000029	CHAR	IOS_SPACEFILE	3-0000010E	CHAR	IOS_SPACERECORD
3-000003D7	CHAR	IOS_STARTDATA	3-000000B4	CHAR	IOS_STARTDATAP
3-00000037	CHAR	IOS_STARTMPROC	3-0000020F	CHAR	IOS_STARTSPNDL
3-00000059	CHAR	IOS_STOP	3-0000000D	CHAR	IOS_UNLOAD
3-0000046B	CHAR	IOS_WRITEBUFNCRC	3-0000011E	CHAR	IOS_WRITECHECK
3-000001E4	CHAR	IOS_WRITECHECKH	3-000003FF	CHAR	IOS_WRITECSR
3-00000153	CHAR	IOS_WRITEHEAD	3-000002A2	CHAR	IOS_WritelBLK
3-00000247	CHAR	IOS_WRITEMARK	3-00000314	CHAR	IOS_WRITEOF
3-0000012A	CHAR	IOS_WRITEPBLK	3-000001C9	CHAR	IOS_WRITERET
3-0000017E	CHAR	IOS_WRITETRACKD	3-00000326	CHAR	IOS_WRITEVBLK
3-00000448	CHAR	IOS_WRITEWTHBUF	3-00000257	CHAR	IOS_WRTTMKR
AP-00000004a	L*1	LUN	3-000004A1	CHAR	QIO_STRING

ARRAYS

Address	Type	Name	Bytes	Dimensions
2-00000000	I*4	QIOCODE	512	(0:1, 0:63)

LABELS

Address	Label
**	10

FUNCTIONS AND SUBROUTINES REFERENCED

Type	Name	Type	Name
	IRP\$W_FUNC	I*4	LIB\$EXTZV

COMMAND QUALIFIERS

FORTRAN /LIS=LIS\$:RXDISK/OBJ=OBJ\$:RXDISK MSRC\$:RXDISK

/CHECK=(NOBOUNDS,OVERFLOW,NOUNDERFLOW)

/DEBUG=(NOSYMBOLS,TRACEBACK)

/STANDARD=(NOSYNTAX,NOSOURCE FORM)

/SHOW=(NOPREPROCESSOR,NOINCLUDE,MAP)

/F77 /NOG_FLOATING /I4 /OPTIMIZE /WARNINGS /NOD_LINES /NOCROSS_REFERENCE /NOMACHINE_CODE /CONTINUATIONS=19

16^{L 15}-Sep-1984 00:14:14
5-Sep-1984 14:21:51

VAX-11 FORTRAN V3.4-56 Page 15
DISK\$VMSMASTER:[ERF.SRC]RXDISK.FOR;1

```
Run Time:          9.73 seconds
Elapsed Time:      23.45 seconds
Page Faults:       249
Dynamic Memory:    235 pages
```


0153 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

